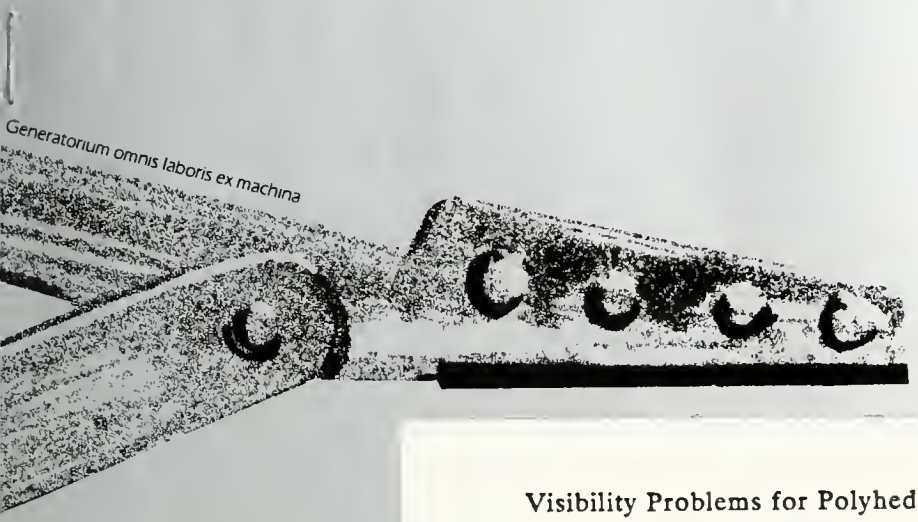
Robotics Research Technical Report



Visibility Problems for Polyhedral Terrains

by

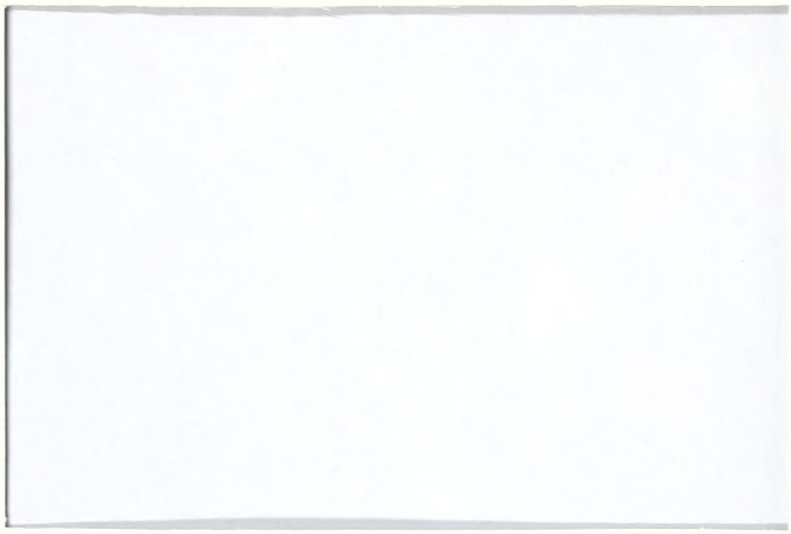
Richard Cole
Micha Sharir

Technical Report No. 92
Robotics Report No. 266
December, 1986

NYU COMPSCI TR-266 c.1
Cole, Richard
Visibility problems for
polyhedral terrains

New York University
Courant Institute of Mathematical Sciences

Computer Science Division
251 Mercer Street New York, N.Y. 10012



Visibility Problems for Polyhedral Terrains

by

Richard Cole
Micha Sharir

Technical Report No. 92
Robotics Report No. 266
December, 1986

New York University
Dept. of Computer Science
Courant Institute of Mathematical Sciences
251 Mercer Street
New York, New York 10012

Work on this paper has been supported by Office of Naval Research Grant N00014-82-K-0381, National Science Foundation CER Grant DCR-83-20085, and by grants from the Digital Equipment Corporation and the IBM Corporation. Work by the first author has also been supported by NSF grant DCR-84-01633 and by an IBM faculty development award.

Visibility Problems for Polyhedral Terrains

Richard Cole[1] *and Micha Sharir*[1,2]

[1] Courant Institute of Mathematical Sciences

New York University

and

[2] School of Mathematical Sciences

Tel Aviv University

ABSTRACT

In this paper we study several problems concerning the visibility of a polyhedral terrain σ from a point (or several points) lying above it. Our results are: (1) For a fixed viewing point a, one can preprocess σ in time $O(n\alpha(n)\log n)$, to produce a data structure of size $O(n\alpha(n)\log n)$, which supports fast *ray shooting* queries, where each such query asks for the point on σ that is visible from a in a specified direction. Here n is the number of faces of σ and $\alpha(n)$ is the extremely slowly growing functional inverse of Ackermann's function. (2) If the viewing point a can vary along a fixed vertical line L, then the entire *visibility structure* of σ from L is of combinatorial complexity $O(n\lambda_4(n))$, where $\lambda_4(n)$ is the maximal length of an $(n,4)$ Davenport-Schinzel sequence, and is nearly linear in n, and where the visibility structure in question is the decomposition of $L\times S^2$ into maximal connected regions, such that for each such region R, all points $(a,u)\in R$ are such that the ray from $a\in L$ in direction $u\in S^2$ first intersects σ at a point on the same face of σ. Furthermore, we present an $O(n\lambda_4(n)\log n)$-time algorithm that preprocesses L and σ into a data-structure of size $O(n\lambda_4(n))$ which supports $O(\log^2 n)$ time ray shooting queries. (3) Concerning the results in (2) we show that (i) if L is not vertical, then the resulting visibility structure can be of size $\Omega(n^3)$; (ii) there exist a vertical line L and a polyhedral terrain σ with n

Work on this paper has been supported by Office of Naval Research Grant N00014-82-K-0381, National Science Foundation Grant No. NSF-DCR-83-20085, and by grants from the Digital Equipment Corporation, and the IBM Corporation. Work by the first author has also been supported by NSF Grant DCR-84-01633 and by an IBM faculty development award.

faces, for which the resulting visibility structure is of size $\Omega(n^2\alpha(n))$. (4) Finally we consider the problem of placing on the surface σ one or several viewing points which collectively cover the entire surface (i.e. each point on σ is visible from at least one of these viewing "stations"). We show (i) in the case of a single viewing station, one can determine in time $O(n \log n)$ whether such a station exists, and if so produce such a point; (ii) the problem of finding the smallest number of points on σ that can collectively see the entire surface σ is *NP*-hard.

1. Introduction

In this paper we consider the variety of 3-D visibility problems mentioned in the abstract, and develop efficient techniques for their solutions, as outlined above. These problems can be viewed as simple 3-dimensional generalizations of planar visibility problems. For example, Chazelle and Guibas [CG2] (cf. also [GHLST]) consider several ray shooting problems for points lying inside a simple polygon P. The results in [CG2] and in [GHLST] show that, after triangulation, P can be preprocessed in linear time to support logarithmic-time ray shooting queries in which the point from which the ray emerges, as well as the direction of the ray, can vary from one query to another.

A direct generalization of this result would aim to preprocess a simple polyhedron σ so as to support fast ray shooting queries from arbitrary interior points in arbitrary directions (these are queries in which the first intersection of such a ray with the boundary of σ is sought). This problem however seems to be much harder than the two-dimensional case if efficient preprocessing time and space are desired. To obtain a preprocessing procedure whose complexity resembles that of the solution in the two-

dimensional case, the problem has to be restricted considerably. The first result of this paper, described in Section 2, is an efficient preprocessing technique for the case in which σ is a polyhedral terrain (i.e. a polyhedral surface having exactly one intersection with each vertical line), and the origin of the query rays is fixed at some point a lying above σ. In this case we obtain an algorithm requiring preprocessing time and space $O(n\,\alpha(n)\log n)$, which supports $O(\log n)$-time ray shooting queries; here $\alpha(n)$ is the extremely slowly growing inverse Ackermann's function. This is a substantial improvement over a naive approach of calculating a decomposition of the space of all rays emerging from a (a space conveniently represented by the unit sphere S^2) into connected regions, each consisting of rays all hitting first the same face of σ. Indeed, it is well known that such a decomposition can consist in the worst case of $\Omega(n^2)$ regions (see Fig. 1.1 for a simple example of this kind).

This result has potential applications to computer graphics and is closely related to the hidden surface removal problem (see [MK], [De]). It can be used to provide compact representation of an image of a polyhedral terrain σ as viewed from some point, from which the actual pixel-by-pixel image can be quickly generated, or the portion of σ appearing at certain "query pixels" can be determined efficiently without having to generate the entire image.

When the origin a of the shooting rays is allowed to vary, the problem becomes harder (even if σ is still assumed to be a polyhedral terrain, rather than an arbitrary polyhedron), and we have not been able to obtain a comparably efficient preprocessing procedure for these extended problems. In Section 3 we consider such an extension, in which the point a can vary along

some fixed vertical line L. Let us denote by the *visibility structure* of σ from L the decomposition of $L \times S^2$ into maximal connected regions; for each such region R, for all points $(a,u) \in R$, the ray from $a \in L$ in direction $u \in S^2$ first intersects σ at a point on the same face of σ. As stated in the abstract we show that the combinatorial complexity of this visibility structure is at most $O(n\lambda_4(n))$, where $\lambda_4(n)$ is the maximal length of an $(n, 4)$ Davenport-Schinzel sequence, and is thus nearly linear in n. See [HS], [Sh1], [Sh2] for details about Davenport-Schinzel sequences and their analysis.

We also present an $O(n\lambda_4(n) \log n)$-time algorithm that preprocesses L and σ into a data-structure of size $O(n\lambda_4(n))$ which supports $O(\log^2 n)$ time ray shooting queries.

To supplement the results just mentioned we show that

(i) If L is not vertical, then the resulting visibility structure can be of size $\Omega(n^3)$.

(ii) There exist a vertical line L and a polyhedral terrain σ with n faces, for which the visibility structure of $L \times S^2$ is of size $\Omega(n^2\alpha(n))$.

By letting the viewing point a vary along more general domains L, and by replaving σ by an arbitrary polyhedral scene, we obtain a collection of progressively more complex versions of the 3-D visibility problem. Some of these generalized problems, in which the viewing point lies at infinity, are discussed in [PD].

Finally we consider in Section 4 the problem of placing one or several viewing points on the surface σ, which collectively cover the entire surface

(i.e. each point on σ is visible from at least one of these viewing "stations"). Such visibility problems arise in many applications, and can be thought of as 3-D generalizations of the well known problems of stationing guards in art galleries [Ch]; see [DFPAN] for a survey of applications of this kind.

We show that

(i) In the case of a single viewing station, one can determine in time $O(n \log n)$ whether such a station exists, and if so produce such a point.

(ii) The problem of finding the smallest number of points that can collectively see the entire surface σ is *NP*-hard.

2. Visibility of a polyhedral surface from a fixed point

Let σ be a 2-D polyhedral surface in E^3, that is σ is the graph of a polyhedral function $z=F(x,y)$, defined over the entire $x-y$ plane, and suppose σ has n edges (and thus $O(n)$ vertices and faces). Let $a=(x_0,y_0,z_0)$ be a fixed point lying above σ (i.e. $z_0>F(x_0,y_0)$). Our goal is to preprocess σ into a data-structure which supports fast responses to *ray-shooting* queries, which ask for the determination of the first point on σ hit by a query ray emerging from a in a specified direction.

Note that each such ray r can be represented by a point $\bar{r} = (\theta,\phi)$ on the unit sphere S^2, where ϕ is the angle between r and the positive z-direction (the "azimuth" of r), and θ is the planar orientation of the horizontal projection of r. Our problem thus calls for calculating a partitioning Π of S^2 such that for each region R of Π, the first intersection with σ by all rays in R emerging from a is at the same face of σ. Once Π is available, we can further

preprocess it using any one of the known algorithms for point location (e.g. that of [DSST]) so as to support efficient (i.e. $O(\log n)$ time) point location queries in Π, which would then correspond to efficient calculation of the face of σ first hit by any query ray. However, as noted in the introduction (cf. also [MK]), the partitioning Π might consist of $\Omega(n^2)$ regions (it always consists of at most $O(n^2)$ regions). Hence, instead of calculating Π explicitly, we want to develop an alternative more space-efficient data-structure that will still support ray shooting queries from a in $O(\log n)$ time.

Our solution proceeds as follows. For any geometric object u in 3-space, denote by u^* its projection onto the $x-y$ plane. The projection σ^* of σ can be regarded as a planar map whose vertices, edges and faces are projections of corresponding vertices, edges and faces of σ. We split σ^* into two disjoint parts σ_1^*, σ_2^* such that each point in σ_1^* lies nearer to the projection a^* of a than any point in σ_2^*. More precisely, define a partial order on the edges of σ^* so that for any pair u, v of edges of σ^*, $u<v$ if there exists a (horizontal) ray emerging from a^* which intersects both u and v such that its intersection with u is nearer to a^* than its intersection with v (see Fig. 2.1(a)). To make this into a partial order we need to ensure that there is some orientation ($\theta = 0$, say) that no edge crosses; this is easily achieved by replacing each edge which does cross $\theta = 0$ with its two subsegments clockwise and counterclockwise from $\theta = 0$. (If we do not have this property there could be a circular sequence of edges $e_1, e_2, \ldots, e_k$, such that $e_i < e_{i+1}$, $1 \le i < k$, and $e_k \le e_1$ (see Fig. 2.1(b)). However, with this extra property it is easy to check that "$<$" is indeed a partial order.) Moreover, it is easy to calculate

this partial order by a standard ray-sweeping procedure in time $O(n \log n)$, and to complete this order into a total linear order in additional $O(n)$ time, using a well known topological sorting procedure. Let this linear order be $e_1 < e_2 < \cdots < e_n$. Split the edges of σ^* into two subsets $\sigma_1^* = \{e_1, \ldots, e_{n/2}\}$ and $\sigma_2^* = \{e_{n/2+1}, \ldots, e_n\}$. Next calculate the *upper rim* of the edges of σ whose projections lie in σ_1^* (call this set of edges σ_1). This is a function $\phi = h(\theta)$ defined over the unit circle S^1, as follows. For each $\theta \in S^1$, let r_θ be the lowest ray that emerges from a, projects onto the horizontal ray at direction θ from a^*, and does not contain any point lying directly below any of the edges in σ_1. $h(\theta)$ is then defined to be the azimuth ϕ of r_θ, i.e. the angle between r_θ and the positive z-direction. Clearly $h(\theta)$ consists of a finite number of smooth connected portions, such that for each such portion γ, all the rays r_θ in γ pass through the same edge of σ_1. The endpoints of these smooth portions are at orientations θ at which either r_θ passes through an endpoint of some edge of σ_1, or r_θ passes simultaneously through two segments of σ_1. We will also use the term "upper rim" to refer to the (not necessarily connected) locus of points on the edges of σ_1 touched by the rays r_θ, for $\theta \in S^1$. See Fig. 2.2 for an illustration of the upper rim of a collection of segments.

Having calculated this upper rim, we then proceed to apply recursively the procedure just described to the two subsets σ_1^* and σ_2^* of the set of edges of σ^*.

The output of this preprocessing is thus a collection of upper rims of subsets of edges of σ, arranged in a balanced binary tree structure T, so that

the root of T stores the upper rim of the half of the edges of σ that lie nearer to a (according to the order defined above), and the left and right subtrees of the root contain the collection of rims calculated recursively for the near half and for the far half of the edges of σ respectively.

Having this structure available, actual ray shooting queries can be processed as follows. Let r be a query ray emerging from a, and let (θ,ϕ) be its spherical coordinates. We perform a binary search through T by first comparing r against the topmost upper rim $h(\theta)$, stored at the root ρ of T; using an auxiliary binary search over θ, we locate the smooth portion of h containing the point $(\theta,h(\theta))$, and then compare (in constant time) ϕ against $h(\theta)$; if $\phi>h(\theta)$ then r lies above all the edges of σ in its nearer portion σ_1, and we continue the search through T at the right child of ρ; otherwise r must hit some face of σ lying in its nearer half, and we continue the search through T at the left child of ρ. When this search is completed, we will have found two edges $e_1<e_2$ which bound the same face f of σ and such that the horizontal projection r^* of r intersects both e_1^*, e_2^* and r lies above e_1 (actually above all edges of σ preceding e_1 whose projections intersect r^*) and below e_2. Hence the first intersection of r with σ lies in f, and can now be calculated in constant time. The whole search takes $O(\log^2 n)$ time.

We can reduce the search time to $O(\log n)$, with only a constant increase in the space requirement, as follows. We use the fractional cascading technique as described in [Co], [CG], [DSST], and elsewhere. The basic idea is to reduce to a constant the search time for going from one node in T to the appropriate child. This is achieved by subdividing smooth portions of upper

rims into smaller angular intervals so that each interval of a rim stored at an internal node overlaps with at most two intervals of the rims stored in each child. It then follows that the total search time can be reduced to $O(\log n)$. Given the original tree T, the modified tree T' can be obtained in time linear in the size of T by a single sweep from the leaves to the root of T, performed level by level. This construction is described in [Co].

The complexity of the approach proposed above crucially depends on the size of the upper rims that it calculates. Fortunately, even though the way in which the entire σ is visible from a can be of $\Omega(n^2)$ combinatorial complexity, upper rims have almost linear size, as shown in the following

Proposition 1: Let $\tau=\{e_1, \ldots, e_n\}$ be a collection of n (non-intersecting) segments in 3-space. Then the graph of the upper rim $h^\tau(\theta)$ of the segments in τ consists of at most $O(n\ \alpha(n))$ smooth portions, where $\alpha(n)$ is the extremely slowly growing inverse Ackermann's function. Moreover, h^τ can be calculated in $O(n\ \alpha(n) \log n)$ time.

Proof: As above we assume that no segment e_l is such that its horizontal projection e_l^* crosses the horizontal ray at orientation $\theta = 0$ from a^*. As noted above, this involves no real loss of generality since we can always replace τ by another collection having at most $2n$ edges which satisfies this additional requirement and which has the same upper rim as τ. Let the sequence of smooth portions of h^τ be $(h_1, \ldots, h_m)$, arranged in counterclockwise order along S^2 and starting at $\theta = 0$, such that for each $i \leq m$ there exists a segment $e_{u_i} \in \tau$ such that all the rays from a having spherical coordinates

$(\theta, h(\theta)) \in h_l$

pass through the segment e_{u_l}. (If in such a smooth portion all these rays pass through two segments e, e' of τ, which can happen if e, e', and a are all co-planar, then we take e_{u_l} to be the farther of these two segments.) Let U be the corresponding sequence $(u_1, u_2, \ldots, u_m)$. We claim that U is an $(n, 3)$-Davenport-Schinzel sequence in the terminology of [HS]. That is, U satisfies the following properties

(1) $u_l \neq u_{l+1}$ for each $i < m$.

(2) For each pair of segment indices $p \neq q$, there do not exist five indices $i_1 < i_2 < i_3 < i_4 < i_5$ in U such that $u_{i_1} = u_{i_3} = u_{i_5} = p$ and $u_{i_2} = u_{i_4} = q$.

Indeed, the first property is obvious. To establish the second property, let $e_p \neq e_q \in \tau$ be such that U contains an alternation of the form $p \cdots q \cdots p \cdots q \cdots p$, and let $\theta_1 < \theta_2 < \theta_3$ be three orientations that correspond to the three middle occurrences $q \cdots p \cdots q$ in this subsequence; in particular, the rays $r_{\theta_1} = (\theta_1, h(\theta_1))$, $r_{\theta_3} = (\theta_3, h(\theta_3))$ pass through e_q, and the ray $r_{\theta_2} = (\theta_2, h(\theta_2))$ passes through e_p. It is easily checked that r_{θ_1} and r_{θ_3} both pass above e_p (in the strong sense that the projection e_p^* intersects the projections $r_{\theta_1}^*$ and $r_{\theta_3}^*$ of these rays, and that above each of these points of intersection the corresponding ray lies above e_p) and that r_{θ_2} passes above e_q. But then it follows by continuity that there must exist two rays r', r'' (whose projections onto the $x-y$ plane have orientations lying in (θ_1, θ_2) and in (θ_2, θ_3) respectively), such that both r' and r'' pass through e_p and through e_q. This implies that e_p, e_q and a must be co-planar, in which

case, by our convention, only one of e_p, e_q should be used to represent any of the smooth portions of h^τ in which it passes through both these segments, a contradiction which establishes our claim.

U is thus an $(n, 3)$ Davenport-Schinzel sequence, and by the results of [HS] the length of U is at most $O(n\,\alpha(n))$. The techniques in [At], [HS] can then be used to calculate h^τ in time $O(n\,\alpha(n)\log n)$, using a straightforward divide-and-conquer approach. □

Using Proposition 1, the complexity of the preprocessing of σ can be estimated as follows. We note that if τ is a collection of segments which is the union of τ_1 and τ_2 then the upper rim of τ can be computed by "merging" the upper rims of τ_1 and τ_2 in time linear in the size of these rims. Thus if we calculate the upper rim of the entire collection τ of the edges of σ, so that at each recursive level we partition the current set of edges into its nearer half and further half, we will obtain, in total time $O(n\,\alpha(n)\log n)$, the entire collection of upper rims that we need to store in our search tree T. (In fact, only half of the rims obtained in this manner need be stored in T; see Fig. 2.3 for an illustration.) The total size of all these upper rims is easily seen to be also $O(n\,\alpha(n)\log n)$. As remarked above, computing the tree T' that supports logarithmic-time queries requires a further $O(n\,\alpha(n)\log n)$ time, so the total time used to build T' is $O(n\,\alpha(n)\log n)$. In summary, we have

Theorem 2: One can preprocess a polyhedral terrain σ having n edges in $O(n\,\alpha(n)\log n)$ time and space, so as to support $O(\log n)$ ray shooting queries from a fixed given point.

Remark: The upper bound of $O(n\,\alpha(n))$ on the complexity of the upper rim of n segments is tight in the worst case. This follows from a recent result of Wiernik and Sharir [WS] that the upper envelope of a collection of n segments in the plane can consist of $\Omega(n\,\alpha(n))$ subsegments. Let $e_1, \ldots, e_n$ be n segments whose upper envelope consists of such a nonlinear number of subsegments, and which are all drawn in the upper half of the $x-z$ plane. We can construct from these segments a polyhedral terrain as follows. Shift each segment e_i in the y-direction to the plane $y = i$, and associate with each shifted segment e_i^* a sharp wedge having e_i^* as its upper edge. The terrain σ is then defined as the upper envelope of all these wedges and of the $x-y$ plane; see Fig. 2.4. Plainly σ has $O(n)$ faces. If we view σ from a point a lying on the y-axis sufficiently far away from these segments, then it is easily checked that the complexity of the resulting upper rim (of the entire collection of segments of σ) is also $\Omega(n\,\alpha(n))$.

3. Visibility of a polyhedral surface from a varying viewing point

In this section we consider an extension of the problem studied in Section 2, in which the point a from which the surface σ is viewed is allowed to vary. We treat here only the simplest extension in which a can be positioned anywhere along a fixed vertical line L. In this case the space of rays that need be processed is 3-dimensional, and is conveniently represented by $L \times S^2$, so that each point $(a,u) \in L \times S^2$ represents the ray emanating from a in direction $u \in S^2$.

The fact that L is vertical enables us to obtain a preprocessing algorithm

whose time complexity is $O(n\lambda_4(n)\log n)$; it produces a data structure of size $O(n\lambda_4(n))$ which supports ray shooting queries in time $O(\log^2 n)$. Our preprocessing algorithm actually yields an explicit representation of the way in which σ can be viewed from L. That is, it produces a decomposition of $L\times S^2$, which we call the *visibility structure* of σ from L, into maximal connected cells, so that with each cell c there is associated a face f_c of σ such that all rays $(a,u) \in c$ first hit σ at a point on f_c. We show that in the worst case the boundaries of the cells in such a decomposition can contain $\Omega(n^2\alpha(n))$ vertices and edges, so that our algorithm is close to being optimal among algorithms required to calculate the entire visibility structure. There remains of course an intriguing open problem whether one can obtain, as in Section 2, a more efficient implicit representation of the visibility structure, that will still support fast ray shooting queries. We also show that if L is not vertical, there are cases in which the complexity of the visibility structure of $L\times S^2$ can be $\Omega(n^3)$, indicating that visibility from a vertical line is indeed a special favorable case.

Our algorithm proceeds as follows. Since all points $a \in L$ have the same projection a^* on the $x-y$ plane, the order among the edges of σ as defined in Section 2 is fixed and independent of the choice of $a \in L$. Our approach is to calculate a search structure, different from that used in Section 2, which is defined as follows.

Consider the collection of all rays in $L\times S^2$ whose horizontal projection has some fixed orientation θ_0. All these rays lie in the vertical half plane $V(\theta_0)$: $\theta=\theta_0$. This plane intersects σ in a polygonal line $\sigma(\theta_0)$, whose

corners lie on edges of σ and whose edges lie on faces of σ. Let $T = T(\theta_0)$ be the collection of (all straight segments comprising) the shortest paths from a^* to all corners of $\sigma(\theta_0)$ constrained to lie in $V(\theta_0)$ above $\sigma(\theta_0)$. It is easily checked (and well known, as a special case of shortest path trees inside simple polygons [GHLST]) that the edges in T form a planar tree rooted at a^*, that there are $O(n)$ such edges, and that the shortest path from a^* to each corner v of $\sigma(\theta_0)$ is concave (i.e. all its bends are downwards). We represent points in $V(\theta_0)$ by the remaining pair (r,z) of their cylindrical coordinates, with z being the vertical axis. See Fig. 3.1 for an illustration of V and T.

Consider now the transformation in which each ray emerging from a point on L in the half-plane $V(\theta_0)$ is mapped into a dual point, as defined in Chazelle and Guibas [CG2] (namely, each ray $z = ar + b$, $r \geq 0$, is mapped to the point $(-a,b)$). As shown in [CG2], in this dual structure, the collection $P(e)$ of all such rays hitting first a fixed edge e of $\sigma(\theta_0)$ is a convex polygon; its vertices are the duals of rays passing through two corners of $\sigma(\theta_0)$ (one of which can be an endpoint of e) before hitting e. Moreover, the special structure of $\sigma(\theta_0)$ implies that each dual polygon $P(e)$ is unbounded. This follows from the fact that if a ray ρ emerging from some $a \in L$ first hits $\sigma(\theta_0)$ at a point $q \in e$, then all rays connecting points $a' \in L$, lying above a, to q have the same property. Let $D(\theta_0)$ denote the partitioning of the dual plane into these polygons; see Fig. 3.2 for an illustration of this structure. Then it is easy to see that as we vary θ slightly near θ_0, the combinatorial structure of the partitioning $D(\theta)$ remains unchanged, unless one of the following critical conditions occur:

(i) $V(\theta_0)$ passes through a vertex of σ.

(ii) Two adjacent edges in $T(\theta_0)$ become collinear.

Call an orientation at which (i) or (ii) occurs a *critical orientation*. Clearly there are only $O(n)$ critical orientations of the first type. To estimate the number of critical orientations of the second type we proceed as follows. For each edge e of σ and for each orientation θ, let $e(\theta)$ denote the intersection of e with $V(\theta)$. Fix an edge e of σ and for each edge $e_i < e$ define the function $f_i(\theta)$ to be the azimuth ϕ of the ray emerging from some point on L and passing through e_i and through the point $e(\theta)$. Define $F_e(\theta) = \max\{ \max_{e_i<e} f_i(\theta) , \phi_0(\theta) \}$, where $\phi_0(\theta)$ is the azimuth of the ray emerging from a^* towards $e(\theta)$. Note that if $F_e(\theta)$ is attained by some f_i (i.e. if $F_e(\theta) > \phi_0(\theta)$), then the last edge in the shortest path from a^* to $e(\theta)$ in $V(\theta)$ (as defined above) must be the segment $e_i(\theta)e(\theta)$. Thus θ is a critical orientation of the second type at which the last edge along the shortest path from a^* to $e(\theta)$ changes if and only if the upper envelope F_e has a "breakpoint" at θ, i.e. F_e is simultaneously attained by two functions f_i, f_j (or f_i, ϕ_0) at θ. For each edge e of σ let $I(e)$ denote the angular sector spanned between the two segments connecting a^* to the two endpoints of e^*. Then it is easy to check that each of the functions $f_i(\theta)$ is partially defined and continuous over the connected angular interval $I(e) \cap I(e_i)$ (and $\phi_0(\theta)$ is defined and continuous over the entire $I(e)$). Moreover, we have

Lemma 3: Each pair of functions $f_i(\theta)$, $f_j(\theta)$ intersect in at most two points. Also each function $f_i(\theta)$ intersects $\phi_0(\theta)$ in at most one point.

Proof: We need to estimate the number of orientations θ at which the three points $e_i(\theta)$, $e_j(\theta)$, and $e(\theta)$ are collinear. Consider the point $e(\theta)$ in the standard (r,z) cylindrical coordinate frame in $V(\theta)$. Let d be the distance from the origin to the line μ containing the projection e^* of e. It is straightforward to check that the (r,z) coordinates of $e(\theta)$ are

$$\left(\frac{d}{\alpha \cos\theta + \beta \sin\theta}, \frac{\gamma \cos\theta + \delta \sin\theta}{\alpha \cos\theta + \beta \sin\theta}\right)$$

for appropriate constants α, β, γ, δ. Similar expressions represent $e_i(\theta)$, $e_j(\theta)$, allowing us to express the collinearity of these three points by an equation of the form

$$\mathbf{DET}\begin{bmatrix} 1 & \frac{d}{\alpha \cos\theta + \beta \sin\theta} & \frac{\gamma \cos\theta + \delta \sin\theta}{\alpha \cos\theta + \beta \sin\theta} \\ 1 & \frac{d_i}{\alpha_i \cos\theta + \beta_i \sin\theta} & \frac{\gamma_i \cos\theta + \delta_i \sin\theta}{\alpha_i \cos\theta + \beta_i \sin\theta} \\ 1 & \frac{d_j}{\alpha_j \cos\theta + \beta_j \sin\theta} & \frac{\gamma_j \cos\theta + \delta_j \sin\theta}{\alpha_j \cos\theta + \beta_j \sin\theta} \end{bmatrix} = 0$$

It is easy to check that this equation can be rewritten as a quadratic homogeneous equation in $\cos\theta$, $\sin\theta$, or rather as a quadratic equation in $\tan\theta$. We thus obtain at most two values of $\tan\theta$, each yielding at most one value of θ in the angular interval $I(e)$ over which F_e is defined (whose length is clearly less than π). This implies that the equation $f_i(\theta) = f_j(\theta)$ has at most two roots. The analysis of equations of the form $f_i(\theta) = \phi_0(\theta)$ is the same, except that the last row of the above determinant is [1, 0, 0] (assuming that a^* lies at the origin). This leads to a linear homogeneous equation in $\cos\theta$, $\sin\theta$ and thus yields at most one root within $I(e)$. □

As shown by Atallah [At], the total number of breakpoints along the

upper envelope of n *partial* functions each having a connected graph, such that each pair of them intersect in at most s points, is $\lambda_{s+2}(n)$. Thus the maximal number of breakpoints in F_e is $\lambda_4(n)$ (which, by the results of [Sh1], is $O(n\,\alpha(n)^{O(\alpha(n))})$). Repeating this argument for each edge e of σ we conclude that the total number of critical orientations of the second type is $O(n\lambda_4(n))$. Furthermore, the divide-and-conquer technique in [At] can be used to obtain all these critical orientations in overall time $O(n\lambda_4(n)\log n)$. Actually, below we will use an alternative technique to calculate these orientations within the same time bound.

We turn to describe the data structure that will be used to represent the visibility structure we seek. First, we define a total order on the faces $f_1, f_2, \ldots, f_n$ of σ so that if $i<j$, then any ray enemating from L and intersecting both f_i and f_j, intersects f_i first. Such an order exists if we make the assumption, similar to that made in the previous section, that no face of σ is cut by the vertical half-plane $\theta = 0$. (To enforce this property, we simply split each face that is cut by that half-plane into two subfaces.) We can then compute this order in linear time as follows. Project the faces of σ onto the $x-y$ plane. Define a partial order between the faces so that $f_i < f_j$ if they are adjacent and all horizontal rays from a^* that meet the edge between f_i^* and f_j^* intersect f_i^* before they intersect f_j^*. Then complete this order to a total order by topological sorting in $O(n)$ time.

Next, we need to study the dual structure $D(\theta)$ more carefully. We will use (r, z) coordinates for the primal half-plane $V(\theta)$, and $(\bar{r}, \bar{z})$ coordinates for the dual structure $D(\theta)$. Consider the faces f_i of σ which are crossed by

the half-plane $V(\theta)$, and let e_i denote the intersection of f_i with that half-plane. To each such segment e_i there corresponds a polygon, which we denote by P_i, in $D(\theta)$. As noted above, P_i is an unbounded convex polygon, and the unbounded rays that it contains all have positive slope. It is also easily checked that the order of the polygons in $D(\theta)$, as they intersect a sufficiently distant horizontal line from right to left, is the same as the order of the corresponding faces f_i of σ. In addition, every bounding edge (of a polygon in $D(\theta)$) consists of dual points of rays that pass through a vertex of $\sigma(\theta)$, so that, arguing as above, each such edge must have positive slope. Thus the line L that separates regions $P_1, P_2, \ldots, P_m$ from regions $P_{m+1}, \ldots, P_n$ is monotone with respect to both the $\tilde{z}$ and the $\tilde{r}$-axes.

We store $D(\theta)$ in a data structure that supports planar point location queries in $O(\log^2 n)$ time, as follows. (There are of course many well-known optimal data structures that will support $O(\log n)$ time point location queries, but it does not appear possible to modify these data structures to allow for varying θ without sacrificing their space efficiency.) We create a complete binary tree Q with n leaves, one for each of the regions $P_1, \ldots, P_n$. We call Q the *region tree*. The symmetric order of the regions in Q is the same as the total order of the corresponding faces of σ. (Since not all faces of σ necessarily intersect $V(\theta)$, some of the regions P_i can be empty for a particular θ.) At each internal node v of Q we store the polygonal line separating the regions stored in the left subtree of v from the regions stored in its right subtree. More precisely, this line consists of those segments which separate a region stored in the left subtree of v from a region stored in the right

subtree. Consequently, this separating line may consist of several unconnected segments. (For example, consider the line separating $P_1, \ldots, P_{n/4}$ from $P_{n/4+1}, \ldots, P_{n/2}$. It might be that at certain points the regions $P_1, \ldots, P_{n/4}$ are adjacent to the line separating $P_1, \ldots, P_{n/2}$ from $P_{n/2+1}, \ldots, P_n$. For such points there is no line separating $P_1, \ldots, P_{n/4}$ from $P_{n/4+1}, \ldots, P_{n/2}$.) Thus each edge in $D(\theta)$ is stored in exactly one separating line; specifically, suppose edge $\tilde{e}$ separates P_i and P_j; let $i \leq k \cdot 2^r < j$ be the integer of this form with the largest possible r; then $\tilde{e}$ belongs to the line separating $P_{(k-1)2^r+1}, \ldots, P_{k \cdot 2^r}$ from $P_{k \cdot 2^r+1}, \ldots, P_{(k+1)2^r}$. Since $D(\theta)$ contains $O(n)$ edges, the overall complexity of all these separating lines is also $O(n)$. Fig. 3.3 shows the collection of separating lines for the structure $D(\theta)$ shown in Fig. 3.2. Each separating line is stored in a balanced binary search tree, ordered by $\tilde{r}$ (or $\tilde{z}$)-coordinate.

We now explain how to perform point location in $D(\theta)$. Suppose that we are locating point (a,b). We start by determining which side of L, the separating line at the root of Q, contains (a,b). We do this by determining the edge e in L that contains a point with $\tilde{r}$-coordinate a, by means of a binary search. We then determine for (a, b) on which side of e it lies, and recursively search the left subtree if (a, b) is below L, or the right subtree if it is above L. In general, there may be no such edge e; but then we know that (a, b) is on the same side of L and of J, where J is the separating line stored at the parent of the current node. Thus we can perform the planar point location in $D(\theta)$ in overall $O(\log^2 n)$ time.

However, we need to produce a data structure which supports fast point location queries in $D(\theta)$ for arbitrary θ. Actually, the data structure we have just described is appropriate for all θ in the range (θ_1, θ_2) where θ_1 and θ_2 are two adjacent critical orientations, as defined earlier. Moreover, when we cross a critical orientation θ of the second type, the structure $D(\theta)$ undergoes only $O(1)$ changes, consisting of addition and deletion of edges; if θ is one of the $O(n)$ type (i) critical orientations, the total change in $D(\theta)$ is at most $O(n)$. This easily implies that the total number of changes in $D(\theta)$, and in the collection of all separating lines, is at most $O(n\lambda_4(n))$.

At each node v of the actual region tree Q, we store the entire collection of separating lines for v, as they vary as θ increases from 0 to 2π, using the similar lists data structure of [Co] or [DSST]. The space requirement for the entire tree Q is thus $O(n + n\lambda_4(n))$, and Q now supports binary search over each separating line in time $O(\log n)$, yielding overall $O(\log^2 n)$ time for a point location query, for arbitrary θ.

It remains to describe how to build the modified data structure Q. This is readily accomplished by sweeping the half-plane $V(\theta)$ from $\theta = 0$ to 2π, maintaining the tree $T(\theta)$ and the dual structure $D(\theta)$, and updating the tree Q at each value of θ in which $T(\theta)$ and $D(\theta)$ change. The initial value of Q is the search structure for the map $D(0)$.

The shortest path tree $T(\theta)$ is stored in a manner consistent with its definition; that is, the parent of each corner v of $\sigma(\theta)$ is its predecessor along the shortest path from a^* to v. The preorder of the nodes of $T(\theta)$ coincides with their r-order as corners of $V(\theta)$. As θ increases, $T(\theta)$ can change in two

ways: (i) Several nodes are removed and replaced by other nodes (this happens at critical orientations of type (i)). (ii) The parent of a node v is changed to the nearest sibling on the left of v or to its grandparent, with the second type of change occurring only if v is the rightmost child of its parent (this happens at critical orientations of type (ii)). Since there are only $O(n)$ changes of type (i), we simply re-calculate $T(\theta)$, in $O(n)$ time, at each of the type (i) critical orientations. Each of the updates of type (ii) is easily accomplished in $O(1)$ time. Thus the total time for all updates of $T(\theta)$ is $O(n\lambda_4(n))$.

To perform these updates during our sweep, we need to know which is the next update to occur. For this, we maintain a heap of candidate critical orientations. In this heap we store the $O(n)$ orientations defined by the $O(n)$ triples of corners of $\sigma(\theta)$ that might cause a change of type (ii), namely: for each pair of adjacent siblings in $T(\theta)$ we store the orientation θ at which the rightmost of the two siblings becomes a child of its sibling, rather than of its parent (i.e the orientation at which the three corners defining the parent and the two siblings become colinear); similarly we store the orientation at which a triple comprising a rightmost sibling, its parent and grandparent become colinear. (For nodes that are children of the root of $T(\theta)$ the second type of change cannot occur, so we do not have to store any corresponding orientation.) We also add to the heap the $O(n)$ critical orientations at which changes of type (i) occur. Note that at each critical orientation of type (ii) there are $O(1)$ updates (insertions and deletions) to be made to the heap; they require $O(\log n)$ time. At each orientation of type (i) we can simply re-calculate the

heap in time $O(n \log n)$. So overall, maintaining the heap takes time $O(n \lambda_4(n) \log n)$.

Finally, we explain how to construct the point location data structure Q. As observed above, each change to $T(\theta)$ produces corresponding changes to $D(\theta)$. Each of these changes consists of the addition or removal of a single known edge from one of the separating lines. Furthermore, if $\bar{e}$ is an edge separating two regions P_i and P_j, which is being added to (or removed from) $D(\theta)$, we can calculate from the indices i and j, in the manner explained above, which node v of Q should store (or already stores) $\bar{e}$. It remains to insert (or delete) $\bar{e}$ into (from) the separating line L stored at v. But this is a standard operation for the similar lists data structure [Co], [DSST], and can thus be done in $O(\log n)$ time. Thus the data structure Q can be built in $O(n \lambda_4(n) \log n)$ time.

To sum up, we have shown

Theorem 4: The visibility structure of a polyhedral terrain σ with n faces in $L \times S^2$, for a given vertical line L, is of size at most $O(n\lambda_4(n))$. It can be calculated in time $O(n\lambda_4(n) \log n)$ and stored into a data structure of size $O(n\lambda_4(n))$ which supports ray shooting queries in $O(\log^2 n)$ time.

Example 5: We do not know whether the bound $O(n\lambda_4(n))$ is tight in the worst case, but the following example shows that the complexity of the visibility structure can be $\Omega(n\lambda_3(n)) = \Omega(n^2\alpha(n))$. In this example we first take, as in Section 2 (and following [WS]), k segments whose upper rim as viewed from a^* consists of $\Omega(k\alpha(k))$ subsegments. Each of these segments

lie in a vertical plane, and these planes are all parallel and sufficiently close to one another so that the combinatorial structure of their upper rim remains unchanged as a varies along L from a^* to a sufficiently high point a^{**}. Moreover, by scaling down the y-coordinate in the construction of [WS], we can assume that all the breakpoints in the upper rim of these segments correspond to rays passing sufficiently near some fixed horizontal line M. We can now complete σ by k additional long horizontal parallel segments lying beyond the first k edges such that the faces they bound form a convex surface (bending upwards), and such that as a varies along L from a^* to a^{**}, the rays shooting from a and passing through M will encounter each of the latter k edges. See Fig. 3.4 for an illustration of such a terrain. It is now clear that for such a terrain σ, the complexity of the visibility structure is $\Omega(n^2\alpha(n))$. In particular, it contains $\Omega(n^2\alpha(n))$ vertices, i.e. rays that pass through three edges of σ simultaneously.

Remark: The requirement that the viewing line L be vertical can be trivially relaxed by requiring only that each line parallel to L intersects σ in precisely one point. In other words, the above analysis will continue to apply for non-vertical lines L, provided that when the coordinate system is rotated so as to make L vertical, σ remains the graph of a single valued polyhedral function.

Example 6: We conclude this section by an example in which the line L is not vertical and the complexity of the visibility structure in $L \times S^2$ is $\Omega(n^3)$. Let L be a horizontal line, and let σ consist of two parallel rows of k sharp peaks each, beyond which k long horizontal parallel edges are placed so that the faces they bound form a convex surface (bending upwards). L is placed in

front of the first row of peaks and parallel to it. See Fig. 3.5 for an illustration of this construction. Now for each pair of peaks p, p', one lying in the first row of peaks and the other in the second row, there exists a point $a \in L$ which is co-planar with two respective parallel sides e, e' of these peaks. Clearly in this plane there exist k rays shooting from a, each of which passes through e, e' and one of the further k horizontal edges of σ. Thus the total number of rays shooting from L and passing through three edges of σ simultaneously is $\Omega(k^3) = \Omega(n^3)$, as claimed. (We are grateful to Ady Wiernik for pointing out this example.) Of course, this does not preclude the possibility of an implicit and more efficient representation of $L \times S^2$ which still supports fast ray shooting queries.

4. Visibility of an entire terrain from several points.

We conclude the paper with a few remarks on the problem of posting several viewing stations on the surface of a polyhedral terrain σ with n faces so that each point on σ is visible from at least one of these stations. We show that for the case of a single station, its existence (and location) can be determined in time $O(n \log n)$ in a straightforward manner. On the other hand, the problem of determining the smallest number of stations that cover the entire terrain is *NP*-hard.

Consider first the case of a single viewing station $a \in \sigma$. For each face f_i of σ let H_i be the (closed) half-space bounded by the plane containing f_i and lying above it. We claim that σ is entirely visible from a if and only if $a \in \bigcap_{i=1}^{n} H_i$. Indeed, for each face f_i, if a can see points on f_i it must clearly lie

in

H_l. Conversely, let $a \in \bigcap_{l=1}^{n} H_l$ and suppose that some point z on a face f_l of σ is not visible from a. Then the ray az must intersect σ. Let b be the last intersection of az with σ before reaching z, and suppose $b \in f_j$. Then it is easily seen that a is not in H_j, a contradiction that proves our claim.

It thus follows that there exists a viewing station that sees the entire surface σ if and only if $\tilde{\sigma} = \sigma \cap (\cap H_l)$ is nonempty, and then any point in $\tilde{\sigma}$ can serve as such a viewing station.

To check whether $\tilde{\sigma}$ is non-empty, and also to find a point on it, we can proceed as follows. First calculate $K = \bigcap_{l=1}^{n} H_l$ in time $O(n \log n)$ using the algorithm in [PM]. Suppose that $\tilde{\sigma}$ is non-empty and that it contains a point a lying in the interior of some face f_l of σ. Let g_l be the face of K co-planar with f_l (g_l may degenerate to an edge or a vertex of K). Then g_l must be contained in f_l. Indeed, if g_l contained a point $b \notin f_l$ then the segment ab must lie entirely on or above σ and must also cross an edge (or vertex) of f_l. But then the face of σ adjacent to f_l across that edge (or some face of σ adjacent to that vertex) will not be visible from a. Similarly, if $\tilde{\sigma}$ contains a point a lying on an edge e_l of σ then the edge of K collinear with e_l must be contained in e_l. This argument proves that $\tilde{\sigma}$ is non-empty if and only if σ contains a vertex of K. This latter property is easy to check using a standard sweeping technique applied to the $x-y$ projection σ^* of σ and to the corresponding projections of the vertices of K, which runs in time

$O(n \log n)$. Thus we have

Proposition 7: Given a polyhedral terrain σ with n faces, one can, in time $O(n \log n)$, determine whether σ contains a point which can see the entire surface σ, and find such a point when it exists.

Consider next the problem of determining the smallest number of viewing stations on σ needed to cover the entire surface of σ (in the sense defined above). We show that this problem is *NP*-hard by reduction from Satisfiability.

Let F be a CNF formula with clauses $C_1, \ldots, C_m$, and variables $x_1, \ldots, x_n$. We reduce the satisfiability problem for F to the problem of determining whether a certain terrain σ with $O(mn)$ faces can be completely viewed by mn points on it.

The surface σ to be viewed is a horizontal plane on which certain *walls* are erected and certain *pits* are dug out. There are $n-1$ parallel walls; they separate *n rows,* one per variable. The pits occur in the rows; There are $2m$ pits per row arranged in a circular fashion as shown in Fig. 4.1, which displays the view of a row from above. The upper rims of the pits are all quadrilateral, and the rims of each pair of adjacent pits in the same row have a common vertex, which we call a *peak*. We create *m columns,* perpendicular to the rows, one per clause. Each column will cut through all the walls. The cross-section of a pit lengthwise (along the row) is shown in Fig. 4.2.

The idea is that only from a peak are the interiors of the two incident pits completely visible, and only from the boundary or interior of a pit can

we see the whole pit. This is achieved by making the pit very deep relative to the height of the walls (e.g. as shown in Fig. 4.3). Thus to see the entire bottom of all the $2m$ pits in a row, at least m viewing points will always be needed. Moreover, let $p_1, \ldots, p_{2m}$ be the peaks in some row r. Then to be able to view all the pits of r with exactly m points, they will have to be placed at every other peak (i.e. either at the odd-numbered peaks or at the even-numbered peaks of that row). Assuming row r corresponds to variable x_r, the choice of even peaks for the viewing points in r will correspond to setting $x_r = \mathit{true}$, while the choice of odd peaks for these points will correspond to setting $x = \mathit{false}$. If clause C_j of our formula does not contain x_r or $\bar{x}_r$ then the peaks p_{2j-1}, p_{2j} in row r are placed outside of any column; if C_j contains x_r (resp. $\bar{x}_r$) then peak p_{2j} (resp. p_{2j-1}) is placed inside the j-th column while p_{2j-1} (resp. p_{2j}) is placed outside of any column. All this can be easily arranged by appropriately varying the lengths of the pits in the r-th row.

A wall has a triangular cross-section (parallel to the columns). Thus a side of a wall can be viewed in its entirety from any peak in either of the adjacent rows. Thus no additional visibility points are required for viewing the sides of the walls parallel to the rows. The cross-section of a wall (parallel to a column) is shown in Fig. 4.4.

Each column makes a vertical cut through each of the walls. The cross section along one wall (parallel to a row) showing the columns is given in Fig. 4.5. Finally, we ensure that each column is completely visible from any peak occurring in the column, but not from any collection of peaks outside

the column. This is achieved by making the walls wide and the columns narrow (the first and last columns do not need to be narrow, and in fact are bounded on just one side). A view from above of the resulting terrain for the formula

$$F = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_3 \vee \bar{x}_4)$$

is shown in Fig. 4.6.

Thus the reduction results in a polyhedral terrain σ having $O(mn)$ faces which requires at least mn viewing points to see all of it. Moreover, if exactly this number of viewing points is used, all of them must be placed at peaks (in order to see all the pits) in the manner described above. Further we must have at least one viewing point per column. But this can occur (with only mn viewing points) if and only if the formula F is satisfiable.

We have thus shown that Satisfiability is reducible to the problem of determining whether a polyhedral terrain σ can be viewed in its entirety from a given number of viewing points; we can therefore conclude

Theorem 8: It is *NP*-hard to determine, for a given polyhedral terrain σ with n faces and a given integer k, whether there exist k viewing stations on σ which collectively see the entire surface of σ.

Remark: Our reduction actually shows that it is *NP*-complete to determine, for a polyhderal terrain σ and an integer k, whether there exist k *vertices* of σ which collectively see the entire surface σ. This latter problem is in *NP* because, after having guessed the k vertices, we can compute in polynomial time the portion of σ visible from each of these vertices, and then check that

the union of these portions is the entire surface σ.

5. Conclusion

In this paper we have analyzed a collection of 3-D visibility problems involving viewing a polehedral terrain from a point or from a collection of points. As far as we know, not much work has been done so far concerning detailed analysis of the computational complexity of these problems, and we hope that the results presented here will spur further progress in this area, which we believe to have many applications in computer graphics, geographical data processing, and other related image processing problems.

Among the immediate open problems that are suggested by our results are:

(1) Preprocess an arbitrary polyhedral surface (not necessarily a terrain), or even an arbitrary collection of disjoint polyhedral surfaces, so as to support fast ray-shooting queries from a fixed viewing point. Can this be done with efficiency comparable to that obtained in Section 2?

(2) Obtain sharp bounds for the combinatorial complexity of the visibility structure for a terrain σ with n faces, when the viewing point a can vary in more general subspaces L of E^3. In particular what is the complexity of this structure when L is the entire half-space lying above σ, or when L is a vertical half-plane lying above σ?

(3) Can the results of Section 3 be improved to obtain implicit and more space-efficient representation of the visibility structure in $L \times S^2$ for a vertical line L?

(4) Recently, Plantinga and Dyer [PD] have considered similar visibility problems concerning parallel views of an arbitrary polyhedral scene from points at infinity, giving a worst-case $O(n^4)$ algorithm for the calculation of the resulting visibility structure. This suggests several open problems, including the problem of obtaining sharp bounds on the complexity of the visibility structure for a polyhedral terrain from points at infinity, and the problem of finding implicit and more space efficient representations of this structure (for terrains as well as for arbitrary polyhedral scenes).

Acknowledgement

The authors wish to thank the participants of the research seminar in computational geometry at the Courant Institute for valuable discussions concerning some of the problems studied in this paper.

References

[At] M. Atallah, Dynamic Computational Geometry, *Proc. 24th Symp. on Foundations of Computer Science,* 1983, pp. 92-99.

[CG] B. Chazelle and L. Guibas, Fractional cascading: A data structuring technique with geometric applications, *Proc. 12th Int. Colloq. on Automata, Languages and Programming,* 1985, Lecture Notes in Computer Science, Springer Verlag, pp. 90-99.

[CG2] B. Chazelle and L. Guibas, Visibility and intersection problems in plane geometry, *Proc. ACM Symp. on Computational Geometry,* 1985, pp. 135-146.

[Ch] V. Chvatal, A combinatorial theorem in plane geometry, *J. Comb. Theory B* 18 (1975) pp. 39-41.

[Co] R. Cole, Searching and storing similar lists, *J. Algorithms* (in press).

[DSST] J. Driscoll, N. Sarnak, D. Sleator and R.E. Tarjan, Making data structures persistent, *Proc. 18th Symp. on Theory of Computing,* 1986, pp. 109-121.

[DFPAN]L. De Floriani, B. Falcidieno, C. Pienovi, D. Allen and G. Nagy, A visibility-based model for terrain features, *Proc. Int. Symp. on Spatial Data Handling,* Seattle, July 1986.

[De] F. Devai, Quadratic bounds for hidden line elimination, *Proc. 2nd ACM Symp. on Computational Geometry,* 1986, pp. 269-275.

[GHLST]L. Guibas, J. Hershberger, D. Leven, M. Sharir and R. Tarjan, Linear time algorithms for visibility and shortest path problems inside simple polygons, *Proc. 2nd ACM Symp. on Computational Geometry,* 1986, pp. 1-12.

[HS] S. Hart and M. Sharir, Nonlinearity of Davenport-Schinzel sequences and of generalized path compression schemes, *Combinatorica* 6 (1986) pp. 175-201.

[MK] M. McKenna, Worst-case optimal hidden-surface removal, Tech. Rept. JHU/EECS-86/05, The Johns Hopkins University, 1986.

[PD] W.H. Plantinga and C.R. Dyer, An algorithm for constructing the aspect graph, *Proc. 27th Symp. on Foundations of Computer Science,* 1986, pp. 123-131.

[PSS] R. Pollack, M. Sharir and S. Sifrony, Separating two simple polygons by a sequence of translations, Tech. Rept. 215, Comp. Science Dept., Courant Institute, April 1986.

[PM] F.P. Preparata and D.E. Muller, Finding the intersection of n half spaces in time $O(n \log n)$, *Theoretical Computer Science* 8 (1979) pp. 45-55.

[Sh1] M. Sharir, Almost Linear Upper Bounds on the Length of General Davenport-Schinzel Sequences, Tech. Report 29/85, The Eskenasy Institute of Computer Sciences, Tel Aviv University, February 1985.

[Sh2] M. Sharir, Improved lower bounds on the length of Davenport Schinzel sequences, Tech. Rept. 204, Comp. Science Dept., Courant Institute, February 1986.

[WS] A. Wiernik and M. Sharir, Planar realization of nonlinear Davenport-Schinzel sequences by segments, Tech. Rept. 224, Comp. Science Dept., Courant Institute, June 1986.

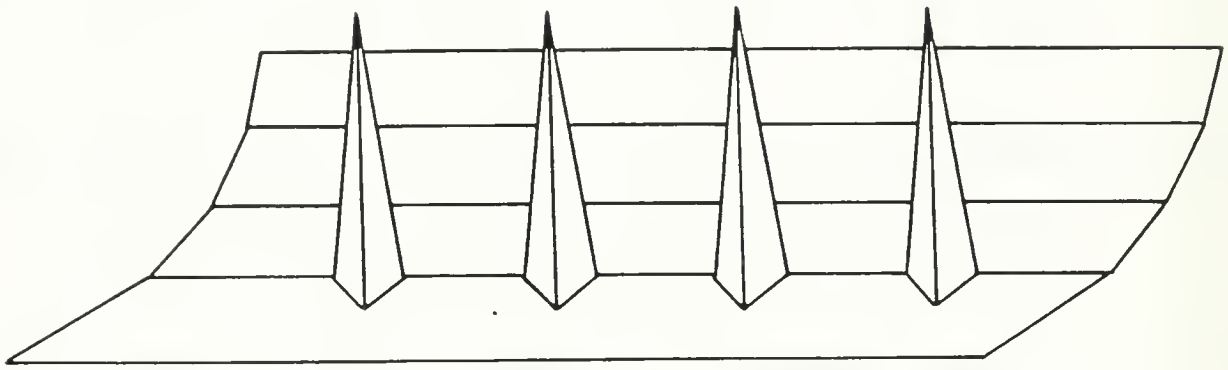
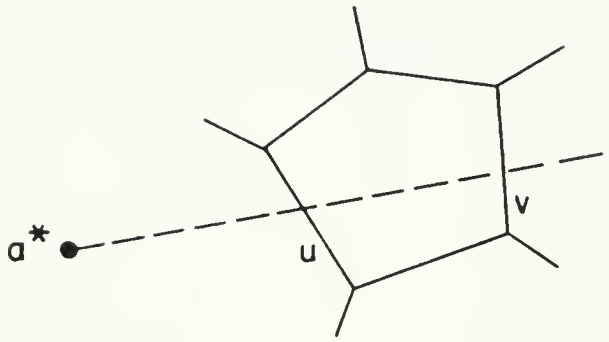
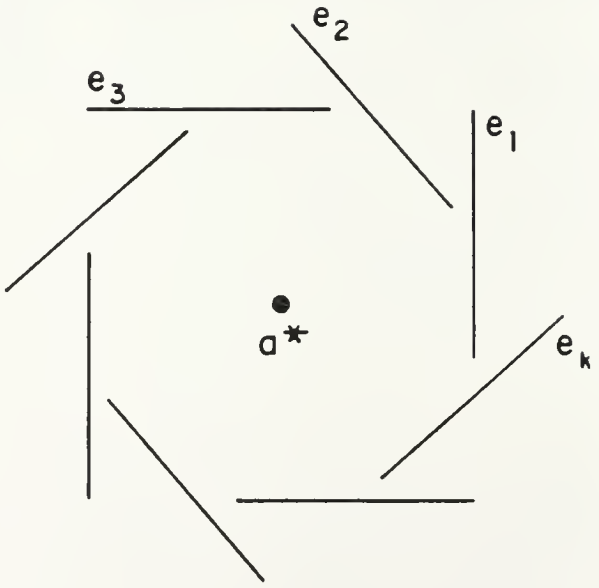


Figure 1.1. Visibility of a terrain from a point with $\Omega(n^2)$ complexity.



(a)



(b)

Figure 2.1(a). The order of edges

Figure 2.1(b). A possible circularity.

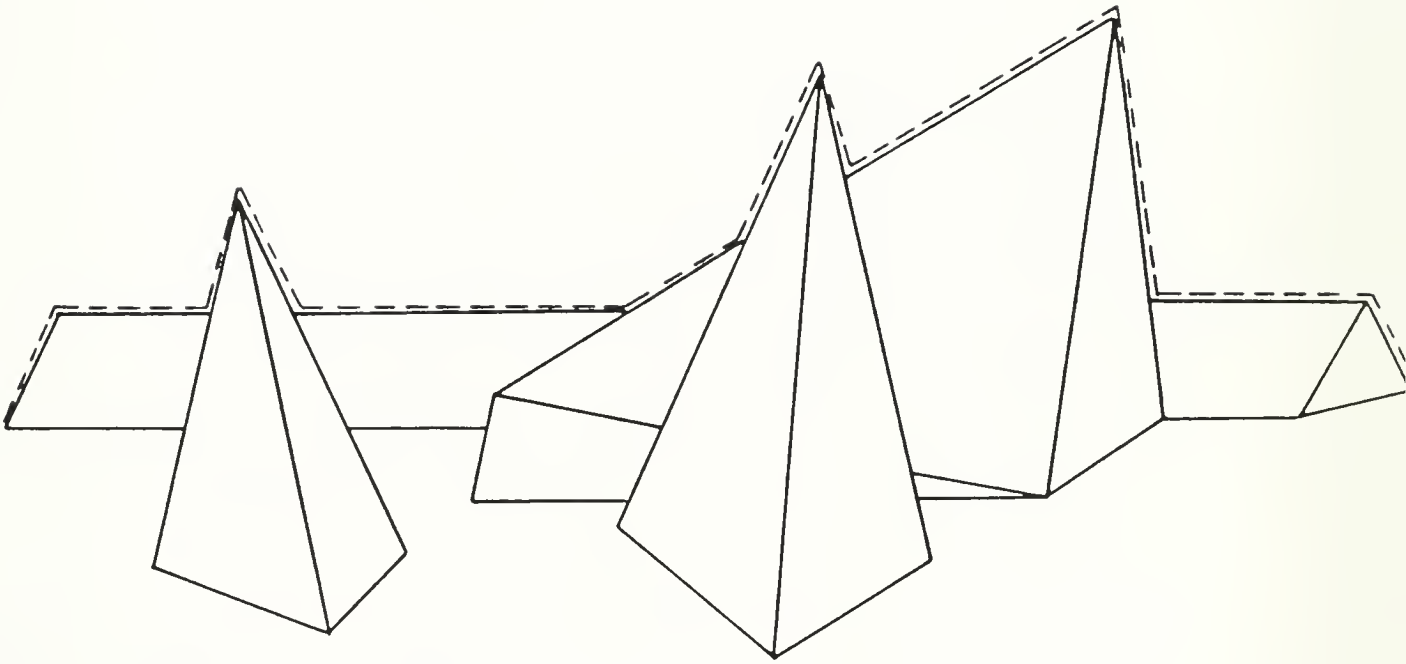


Figure 2.2. The upper rim of a terrain.

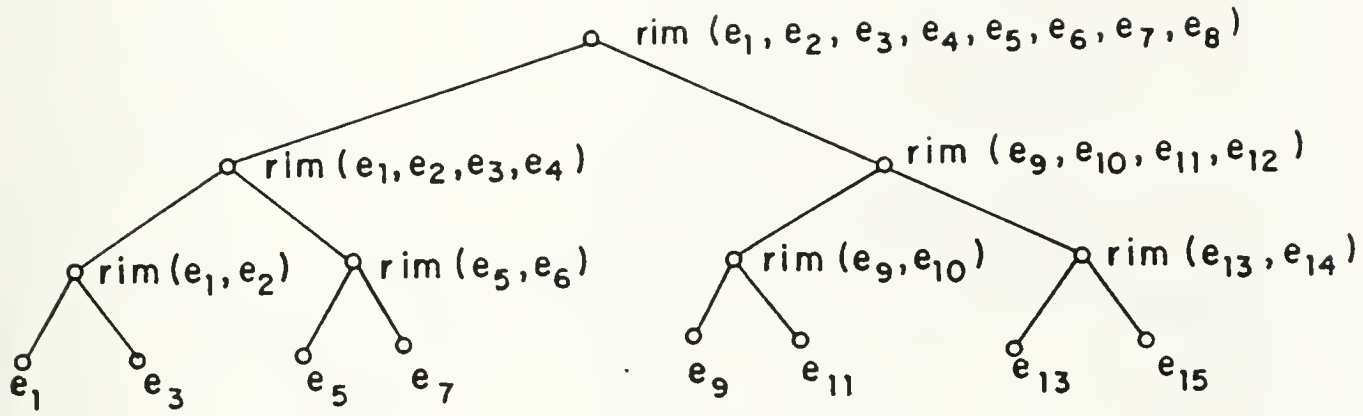


Figure 2.3. The tree of upper rims for a terrain with 16 edges.

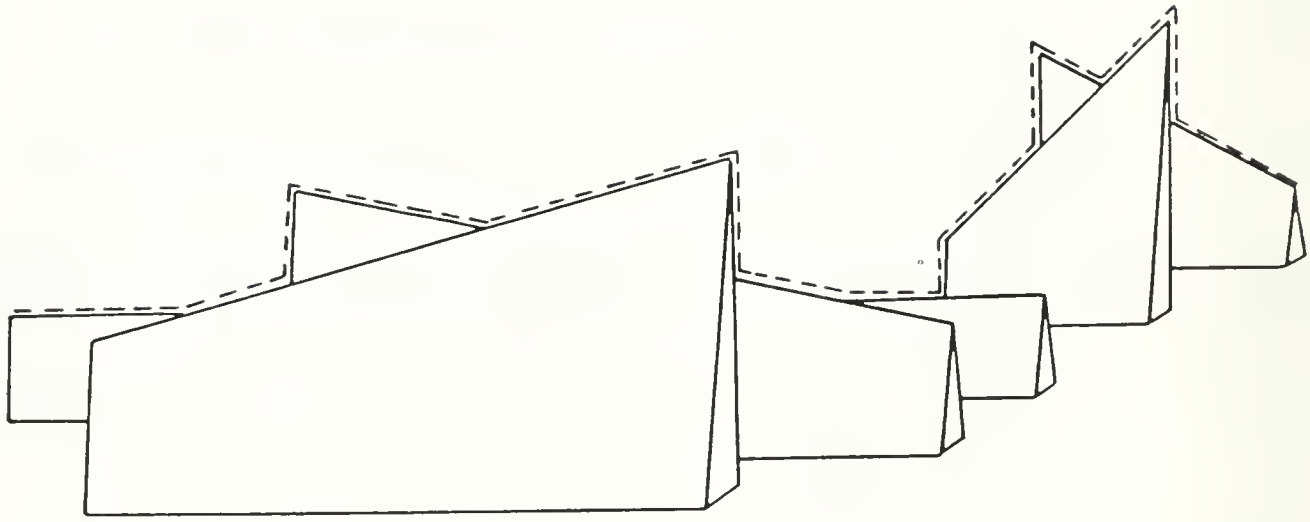


Figure 2.4. The lower bound construction for upper rims.

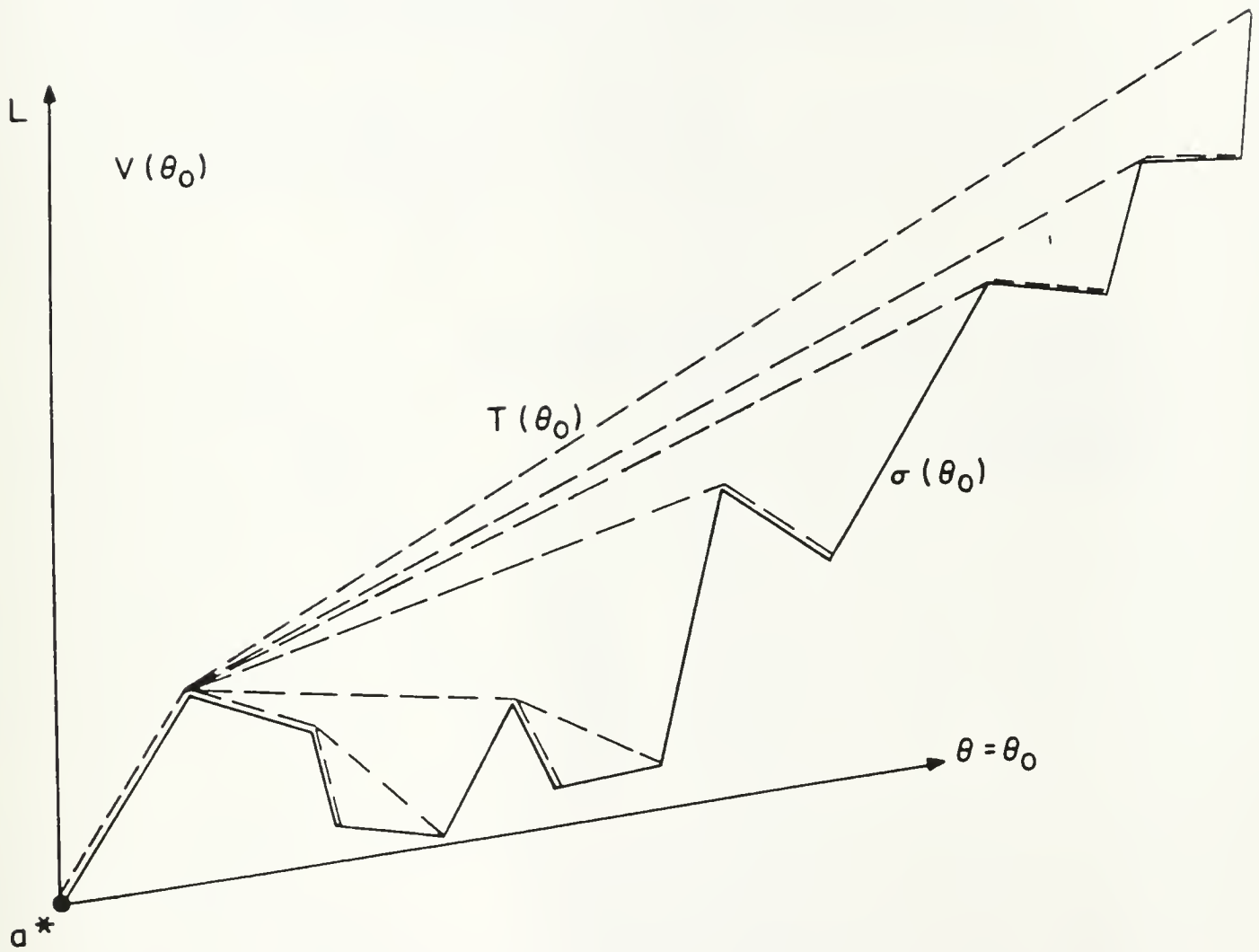


Figure 3.1. $V(\theta_0)$ and $T(\theta_0)$.

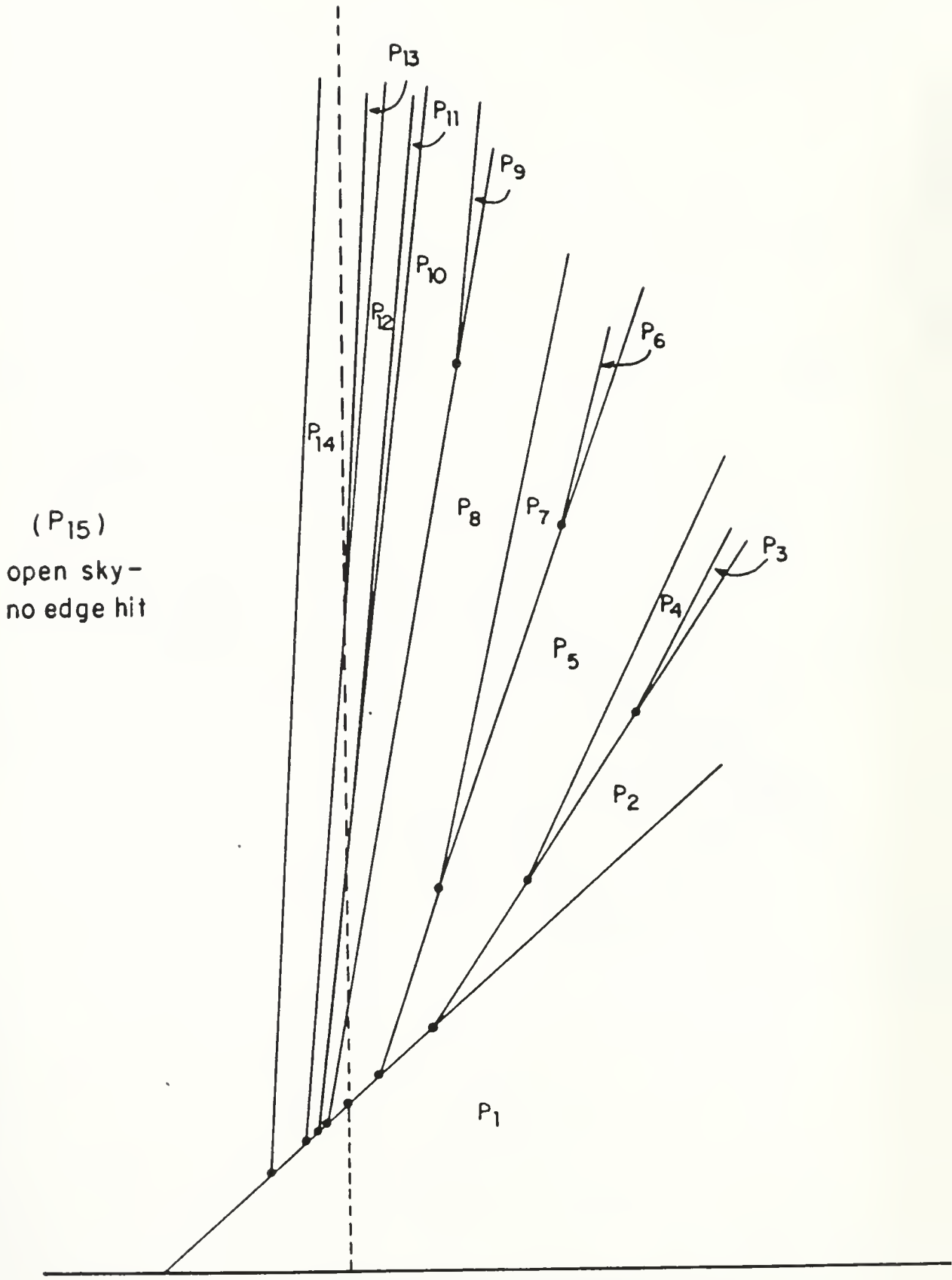
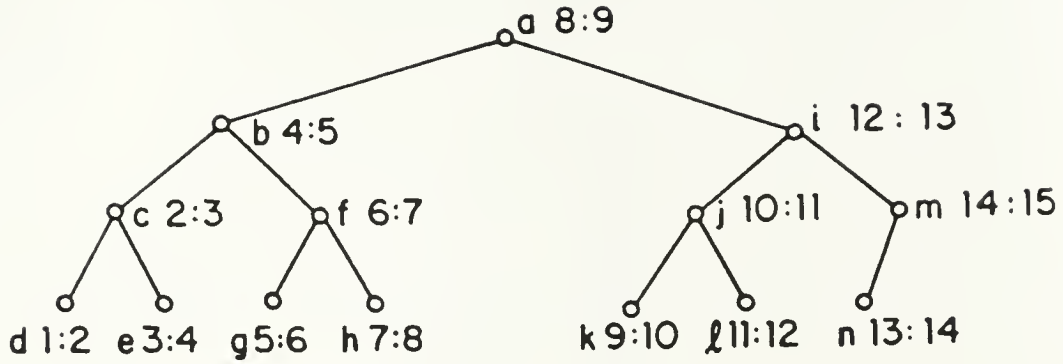


Figure 3.2. The structure $D(\theta_0)$ for the structure of Figure 3.1.



The internal nodes of Q, each labeled by i:i+1, where P_i is the largest-numbered region in the left subtree of the node.

Node of Q	Separating edges stored at the node (i,j) designates the edge separating P_i and P_j
a	(1,15),(1,14),(1,12),(1,10),(8,10),(8,9)
b	(1,8),(1,5),(2,5),(4,5)
c	(2,4),(2,3)
d	(1,2)
e	(3,4)
f	(5,8),(5,7),(6,7)
g	(5,6)
h	(7,8)
i	(12,14),(12,13)
j	(10,12),(10,11)
k	(9,10)
l	(11,12)
m	(14,15)
n	(13,14)

Figure 3.3. The region tree Q and the separating lines.

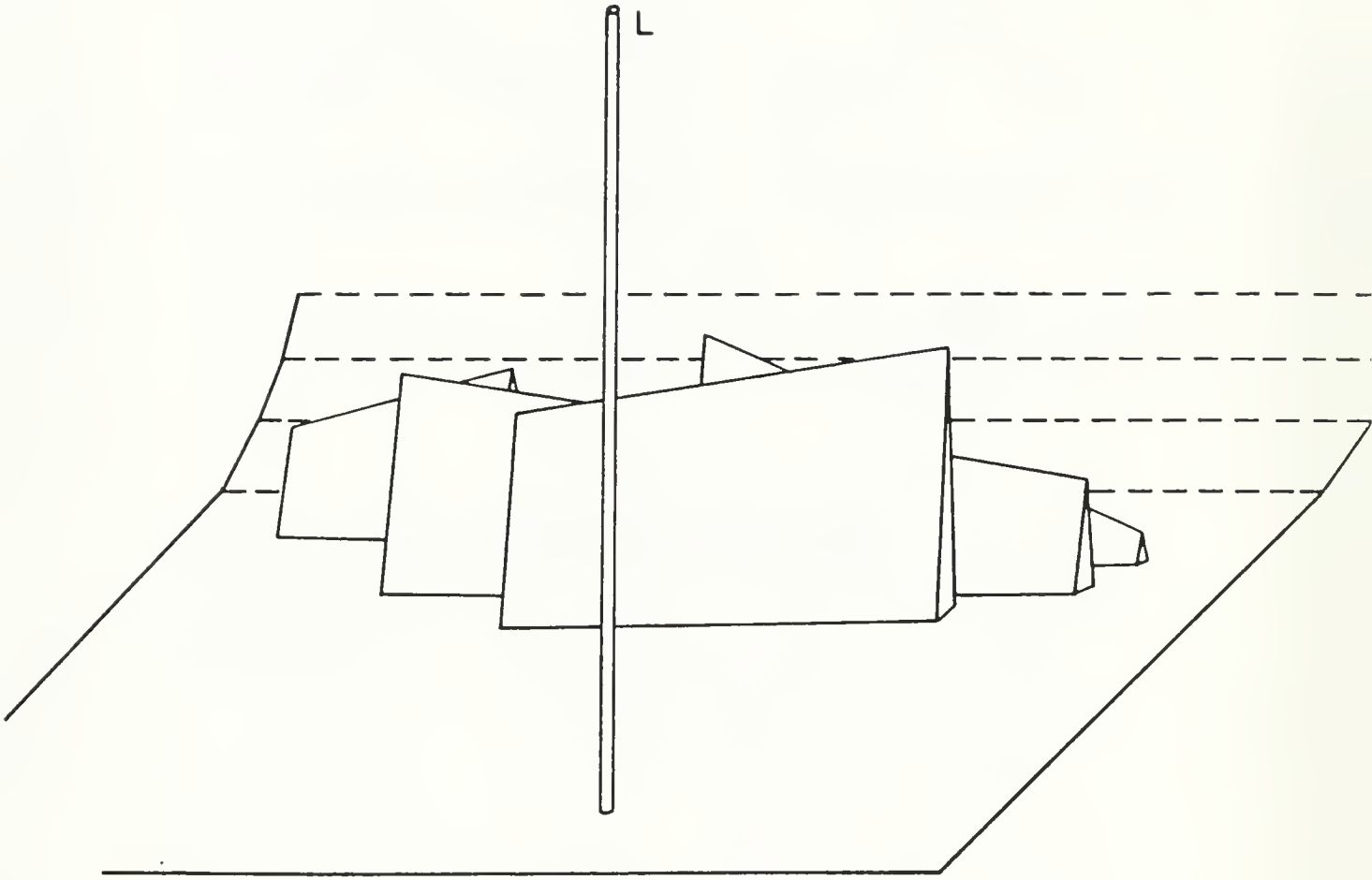


Figure 3.4. The lower bound construction for visibility from a vertical line.

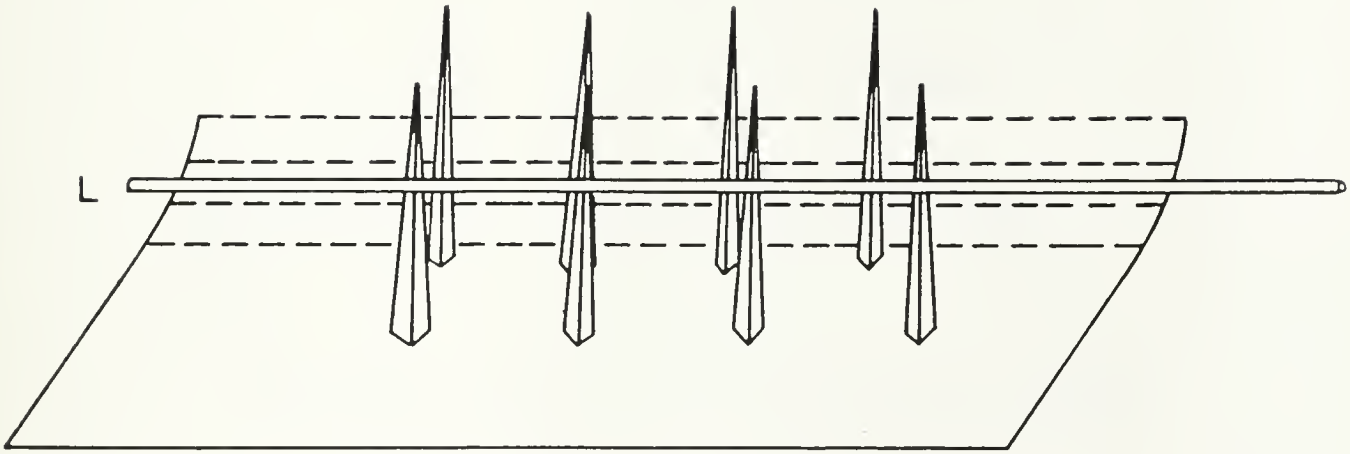


Figure 3.5. The lower bound construction for visibility from a horizontal line.

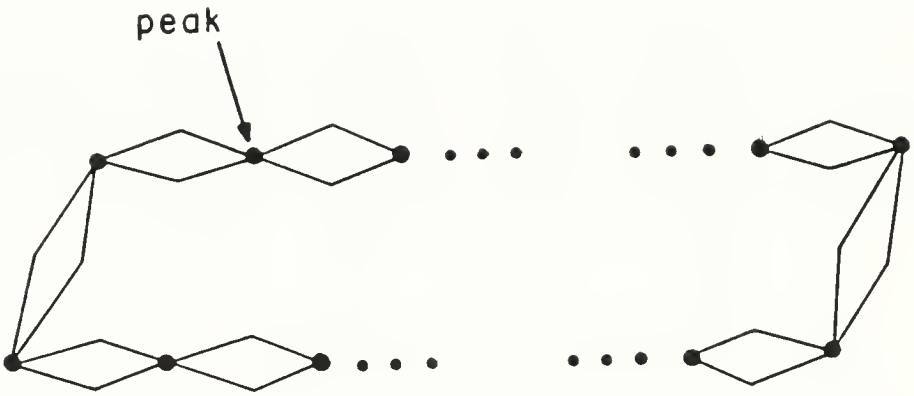


Figure 4.1. Schematic arrangement of pits in a row.

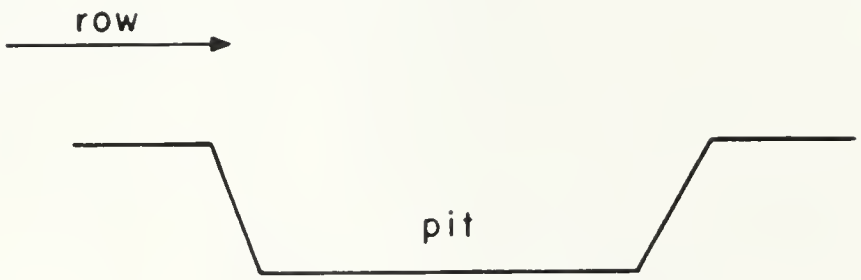


Figure 4.2. A vertical cross-section of a pit.

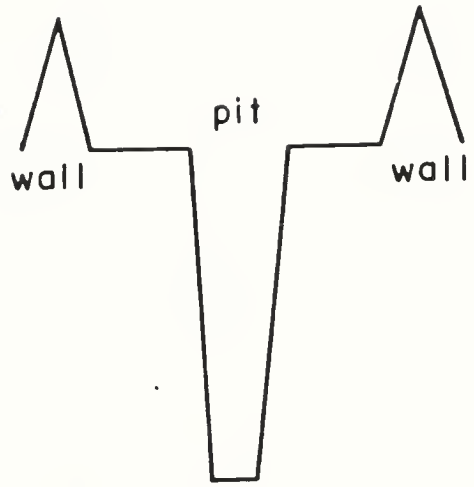


Figure 4.3. A vertical cross-section across a row.

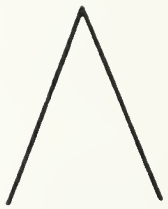


Figure 4.4. A vertical cross-section of a wall.

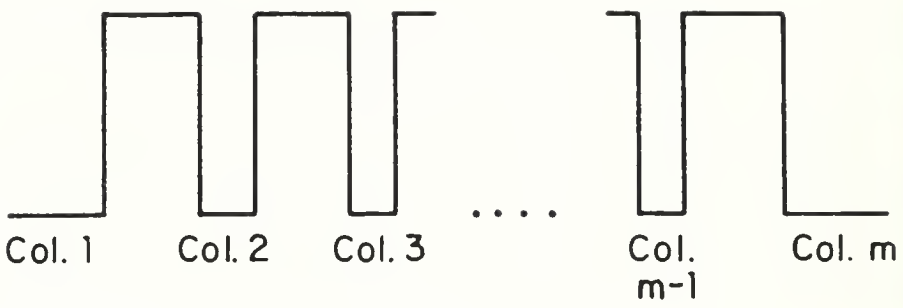


Figure 4.5. A wall with its column appertures viewed from an adjacent row.

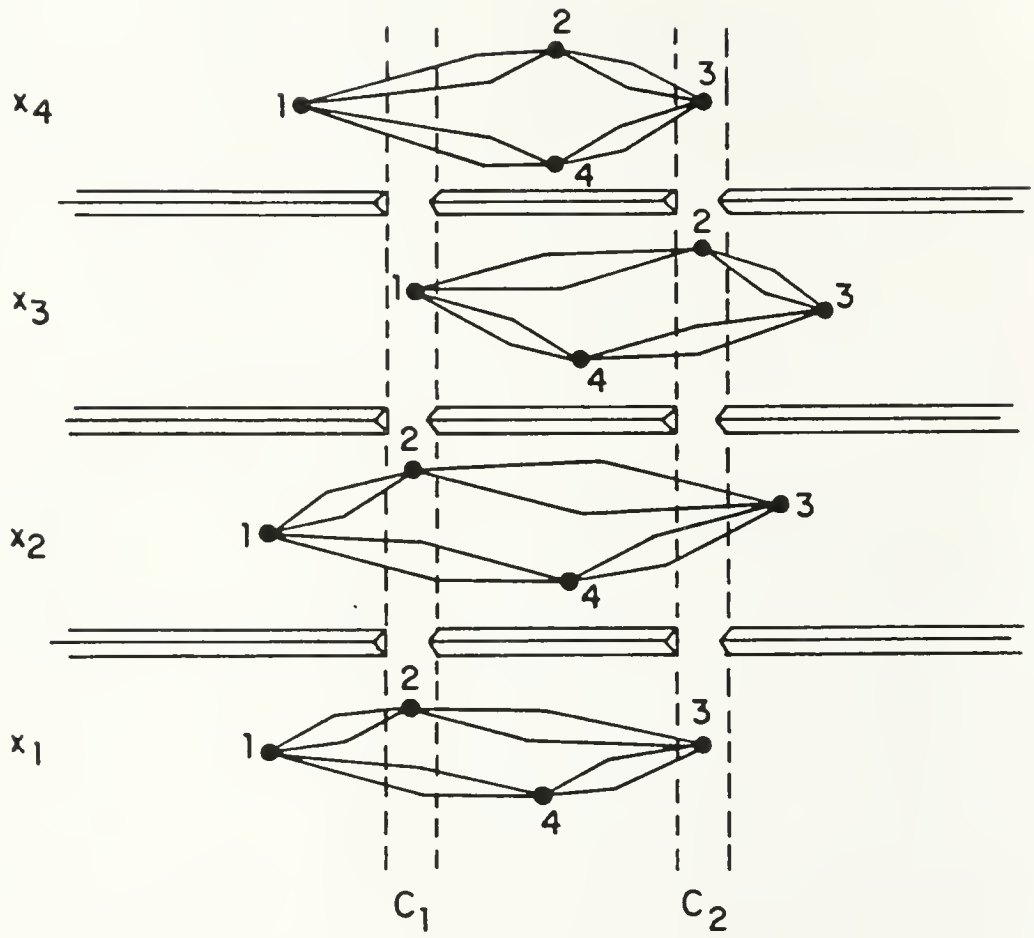


Figure 4.6. A view from above of the terrain constructed for $F = (x_1 \vee x_2 \vee \bar{x}_3) \wedge (\bar{x}_1 \vee x_3 \vee \bar{x}_4)$.

NYU COMPSCI TR-266 c.1
Cole, Richard
Visibility problems for
polyhedral terrains

NYU COMPSCI TR-266 c.1
Cole, Richard
Visibility problems for
polyhedral terrains

MAR 2 5 198
DATE DUE

GAYLORD PRINTED IN U.S.A.